Amiri, S., Hosseinabady, M., McIntosh-Smith, S., & Nunez-Yanez, J. (2018). Multi-precision convolutional neural networks on heterogeneous hardware. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018* (pp. 419-424). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.23919/DATE.2018.8342046

Multi-Precision Convolutional Neural Networks on Heterogeneous Hardware

Moslem Amiri, Mohammad Hosseinabady, Simon McIntosh-Smith and Jose Nunez-Yanez
Faculty of Engineering, University of Bristol, Bristol, UK
Emails: {ma17215, m.hosseinabady, s.mcintosh-smith, j.l.nunez-yanez}@bristol.ac.uk

***Abstract*—Fully binarised convolutional neural networks (CNNs) deliver very high inference performance using single-bit weights and activations, together with XNOR type operators for the kernel convolutions. Current research shows that full binarisation results in a degradation of accuracy and different approaches to tackle this issue are being investigated such as using more complex models as accuracy reduces. This paper proposes an alternative based on a multi-precision CNN framework that combines a binarised and a floating point CNN in a pipeline configuration deployed on heterogeneous hardware. The binarised CNN is mapped onto an FPGA device and used to perform inference over the whole input set while the floating point network is mapped onto a CPU device and performs re-inference only when the classification confidence level is low. A light-weight confidence mechanism enables a flexible trade-off between accuracy and throughput. To demonstrate the concept, we choose a Zynq 7020 device as the hardware target and show that the multi-precision network is able to increase the BNN accuracy from 78.5% to 82.5% and the CPU inference speed from 29.68 to 90.82 images/sec.**

***Index Terms*—multi-precision, performance, Convolutional Neural Network, deep learning, heterogeneous, FPGA, ARM, CIFAR-10, inference**

I. Introduction

Recent advancements in convolutional neural network (CNN) performance has been possible thanks to multi- and many-core hardware platforms with parallel CPU, GPU and FPGA resources. These powerful platforms enable training and inference of deep CNNs with hundreds of object categories and millions of images such as ImageNet [1]. Recent research has shown that reduced precision neural networks with 8 and 16-bit operators can improve the throughput in deep CNNs with minimal impact on accuracy [2]. The extreme case of fully binarised Neural Networks (BNNs) presented in [2] shows a large reduction in the memory requirements and the design compute datapath at a cost of around 13% in accuracy. BNNs are very suitable for FPGA hardware since it is possible to store all their network parameters in internal memory eliminating access to external memory and use custom hardware to build the single-bit operators.

This paper proposes a multi-precision CNN integrating two different CNN implementations in which one of them brings throughput, such as a BNN, and the other provides accuracy, such as a full-precision CNN. The high-throughput implementation is the main inference engine; if the classification confidence of an image is low then the high-accuracy implementation will be activated. A light-weight mechanism is used to make the decision to switch from the high-throughput CNN to the high-accurate CNN. The two networks operate in parallel as a stream of images pass through the system. Fig. 1 depicts this concept.

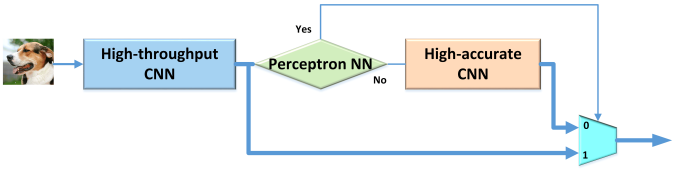

Fig. 1. Overview of the proposed multi-precision CNN.

The contributions of this article are:

- Introduction and analysis of a parallelised multi-precision design composed of two different-precision networks, one based on the binarised FINN network [3] providing throughput and the other a Caffe-based floating-point network providing accuracy;
- Proposing a trainable and adjustable method based on Softmax function to identify the incorrectly-classified images by the high-throughput network and activate the high-accurate network for their re-classification in order to provide throughput/accuracy balance;
- Port of the original binarised FINN network to a Linux-based Zynq compute environment using the Xilinx SDSoC high-level synthesis tools and optimization of the BNN for different utilizations of on-chip memory and hardware resources;
- Classification of CIFAR-10 test dataset in three multi-precision experiments, each with a different floating-point Caffe network combined with the FINN network. A balanced multi-precision experiment, integrating a light floating-point network with FINN, demonstrates that FINN accuracy is improved from 78.5% to 82.5% and floating-point network inference rate is improved from 29.68 to 90.82 images/sec.

II. Previous Work and Background

Floating-point CNNs provide high classification accuracies in multiple application areas but require large compute/memory resources usually available in multi-core and many-core hardware [4]. A number of compression techniques have been

This work was funded by the Engineering and Physical Sciences Research Council (EPSRC) through the ENEAC (EP/N002539/1) grant.

introduced to limit compute and/or memory requirements of neural networks. In [5], a three-stage pipeline compression technique is introduced where pruning, trained quantization and Huffman coding work together to reduce the storage requirement of neural networks. SqueezeNet [6] is a small CNN architecture which uses reduced precision with fixed-point arithmetic and fewer parameters than the full network, hence suitable for deployment on hardware with limited memory. Binarised Neural Networks (BNNs) [2] are the most reduced precision CNNs compressed into only single-bit parameters, improving both the implementation datapath and memory costs.

An efficient framework to implement BNNs on FPGA is FINN [3]. FINN is based on the BNN method developed by Courbariaux et al. [2] providing high performance and low memory cost using XNOR-popcount-threshold datapaths and with all the parameters stored in on-chip memory (OCM). FINN has a streaming multi-layer pipeline architecture where every layer is composed of a compute engine surrounded by input/output buffers. A FINN engine implements the matrix-vector products of fully-connected layers or the matrix-matrix products of convolution operation which is obtained by unrolling the convolution operation [7]. An engine computes binarized products and then compares against a threshold for binarized activation. It consists of *P* processing elements (PEs), each having *S* SIMD lanes processed simultaneously in one clock-cycle. Every layer is scalable rather than fixed and the number of PE and SIMD lanes in the engines can be balanced according to the their compute requirements. The first layer of the network receives non-binarised image inputs hence requiring regular operations, and the last layer outputs non-binarised classification result and does not require thresholding. Non-binarised operations can also be extended to handle inputs and outputs in inner layers resulting in a partially-binarised network.

III. Multi-Precision CNN

As explained in Section I, we propose a multi-precision design which balances the features of different solutions and is built with three components: a CNN with high throughput, a CNN with high accuracy and a light-weight NN as the Decision Making Unit (DMU) in between. In our current configuration, a single-bit BNN network is considered as the high-throughput CNN and a floating-point NN provides the high-accuracy CNN. Using an FPGA-CPU heterogeneous device, we execute the single-bit network on the FPGA and the floating-point network on the CPU. After initial classification of the dataset in the BNN, a trained DMU decides to send a selected subset of images for re-inference to the floating-point network. DMU can be set to different thresholds to adjust accuracy vs. speed by controlling the number of reclassifications in the floating-point network. For this work, the BNN implementation on the FPGA is based on FINN. Using the SDSoC tools (version 2016.3), the original FINN implementation is ported to a Linux environment and a hardware library is generated. The original FINN interface code based on Vivado HLS is removed and SDSoC pragmas are added to guide the

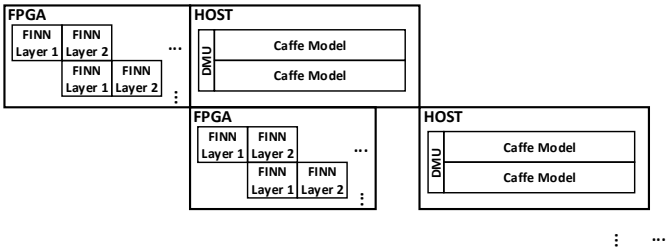

Fig. 2. Pipelined implementation of our multi-precision design on a ZC702 board. Other than the concurrent execution of FPGA with host, FINN layers are also pipelined and dual-core processors run in parallel.

SDSoC cross-compiler with the hardware and Linux driver generation. The result of this process is a bitstream ready to be uploaded in the FPGA and a software library that can then be linked with the rest of the host code. The host code is compiled using standard g++ in the target board directly running Linaro 12.11. The advantage of using these libraries and splitting host and hardware compilation is that the host code can be compiled with the latest g++ version supporting C++11, and the SDSoC environment sds++ compiler only needs to handle the driver and hardware generation. The evaluation board used in our experiments is the Xilinx Zynq-7000 ZC702, featuring the XC7Z020 SoC with Artix-7 programmable logic and a processing system containing a dual core ARM Cortex-A9 processor with a maximum clock frequency of 666 MHz.

The pipeline diagram of the multi-precision design in the FPGA-CPU heterogeneous device is shown in Fig. 2. On every pass, the FINN implementation processes a batch of images in FPGA and the Caffe network classifies a subset of the batch which was previously processed by FINN in the FPGA. The selection of this subset is implemented by the DMU which is executed in the processor.

In order to provide asynchronous execution and parallelism between hardware and host CPU, corresponding to the execution style shown in Fig. 2, the paired SDSoC pragmas of `SDS async(ID)` and `SDS wait(ID)` are used, as illustrated in the following pseudo-code:

```
for (i = 0; i < image_numbers/batch_size; i++){
        #pragma SDS async(1)
        FPGA_execution(batch[i]);
        if (i > 0) ARM_execution(incorrectly
            classified images of batch[i-1]);
        #pragma SDS wait(1)
}
ARM_execution(incorrectly classified images of last
    batch);
```

Using this method, the compiler does not generate the wait after a call to a hardware function; after transferring the inputs to hardware, the program returns immediately and executes to the point of `SDS wait(1)` and then checks the hardware output and waits if it is not ready.

Given the accuracy and speed of the two different precision designs and also the DMU details and settings, the performance of the resulting multi-precision system can be measured. If $t_{\text{fp/img}}$ and $t_{\text{bnn/img}}$ are the times taken to process an image in a floating-point network in host and a single-point network in FPGA respectively, and R_{rerun} is the ratio of

images in a batch re-processed in host ($0 \leq R_{\text{rerun}} \leq 1$), then the average interval time to process an image in the multi-precision system will be

$$t_{\text{multi-prec/img}} \approx \max\{t_{\text{fp/img}} \times R_{\text{rerun}},\, t_{\text{bnn/img}}\}. \quad (1)$$

Since in general the host re-inference latency is the bottleneck, $t_{\text{multi-prec/img}}$ can be adjusted through several factors including the computational load of the network in host, the performance of the host processor, utilisation of a different trained DMU network or a different threshold setting for it; the trade-off resulting from the multi-precision system provides the timing gain of $t_{\text{fp/img}} \times (1 - R_{\text{rerun}})$ for the host processor.

If $R_{\text{rerun\_err}}$ is the ratio of images which have been initially classified correctly by BNN method but due to DMU error are reprocessed by host, and Acc_{bnn} and Acc_{fp} are the accuracies for networks in FPGA and host which are scaled to 0–1 range, the accuracy of the multi-precision system can be measured as

$$Acc_{\text{multi-prec}} \approx Acc_{\text{bnn}} + (Acc_{\text{fp}} \times R_{\text{rerun}}) - R_{\text{rerun\_err}}. \quad (2)$$

The balanced multi-precision system improves the accuracy of the BNN method by $(Acc_{\text{fp}} \times R_{\text{rerun}}) - R_{\text{rerun\_err}}$. For a higher accuracy gain, a deeper host network with higher Acc_{fp} or a DMU with lower $R_{\text{rerun\_err}}$ is required. Although higher R_{rerun} or larger host network might increase the accuracy, it will also increase $t_{\text{multi-prec/img}}$, hence the host network choice or the DMU design and its threshold setting should be dependent on the desired overall requirements. In practice, $Acc_{\text{multi-prec}}$ is lower than the one acquired by (2) as Acc_{fp} drops when a subset of hard-to-classify images are re-inferred in the host.

Changing batch size does not have a significant effect on multi-precision features. From a practical point of view, as batch size increases, there is slightly less overhead in the buffers and FIFOs but a larger pipeline ramp up and ramp down due to the larger batch size. Also with higher batch sizes, the latency of an image to pass through the multi-precision system increases.

In the remaining of this section, the three components of the multi-precision system (fast network, DMU, and accurate network) are introduced and analysed separately and then the results obtained by the combined system are presented. In particular, we focus on the processing of CIFAR-10 classification example[1]. CIFAR-10 dataset contains 32x32 colour images in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). It includes 50000 training images and 10000 test images.

A. FINN Scaling Analysis

The FINN network layers used in this experiment to classify CIFAR-10 dataset are listed in Table I. Each layer of the network is implemented in one engine and every engine has two parameters to control how a weight matrix is partitioned and hence its performance: P which is the number of the processing elements (PEs) of the engine, and S which is the number of SIMD lanes per PE. A $P \times S$ tile of weight matrix is processed at a time where each row of the tile is mapped to a different PE and each column to a different SIMD lane. The performance of every layer in a FINN design is controlled by that layer's P and S, and the worst case layer decides the performance of the whole network. Hence to adjust the design based on the requirements, the features involved should be known. For a convolution or fully-connected (FC) layer of a FINN network with P PEs and S SIMDs, given the following feature sizes:

- Convolution kernel: $K \times K$
- Convolution layer input: $IH \times IW \times ID$
- Convolution layer output: $OH \times OW \times OD$
- FC layer input: ID
- FC layer output: OD

then the following features are obtained for the layer:

- Total weight size of a convolution layer: $OD \times (K * K * ID)$
- Total weight size of a FC layer: $OD \times ID$
- Weight memory: P files each containing $\frac{\text{Total weight size}}{P*S}$ arrays of S-bit values
- Threshold memory: P files each containing $\frac{OD}{P}$ arrays of a constant bit size (in our experiment, 24-bit for first stage and 16-bit for the rest except the last stage.)

With the above specification, the total number of clock cycles (CC) for a convolution layer to produce all its activations will be

$$CC_{CONV} = \frac{OD}{P} * \frac{K * K * ID}{S} * OH * OW \quad (3)$$

and for a FC layer will be

$$CC_{FC} = \frac{OD}{P} * \frac{ID}{S}. \quad (4)$$

For a given layer to complete its computation, the number of clock cycles measured by (3) or (4) agrees with the estimated report provided by Analysis Perspective of Vivado HLS environment; to do this, we set the batch size to 1 since otherwise the amount of processing would be dependent on a run-time value and latencies would be unknown at compile time.

TABLE I
THE FINN ENGINES OF THE NETWORK USED IN OUR EXPERIMENT FOR CLASSIFICATION OF CIFAR-10 (NO ZERO PADDING IS APPLIED)

FINN
input (32×32 RGB image)
3×3-conv-64
3×3-conv-64
pooling
3×3-conv-128
3×3-conv-128
pooling
3×3-conv-256
3×3-conv-256
FC-64
FC-64
FC-64 (no activation)

[1]https://www.cs.toronto.edu/~kriz/cifar.html

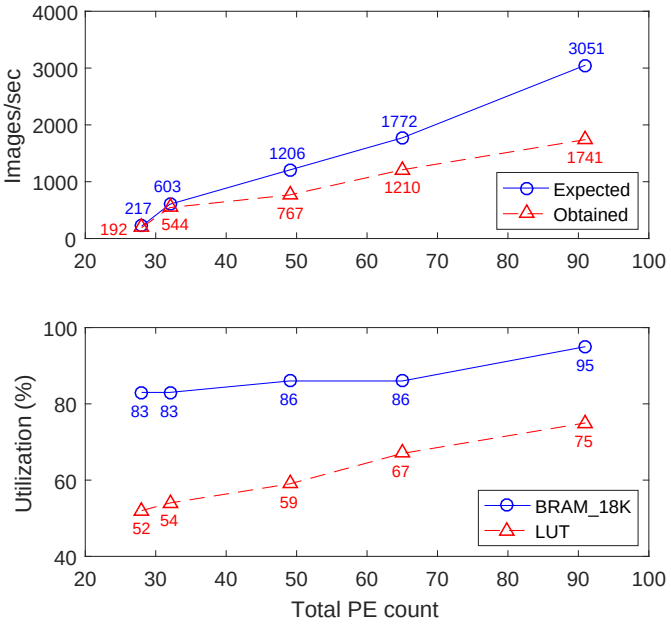

Fig. 3. Performance and area utilization for different CIFAR-10 implementations on ZC702 board each with a different degree of parallelism.

In order to maximise the throughput, it is necessary to rate-balance the heterogeneous streaming network layers. The layer with the highest latency will determine the overall throughput. Since in a layer P and S determine its throughput, for a rough balance of all the layers and given one desired latency (in CC), (3) or (4) should be assessed for each layer to find a combination of P and S for that layer satisfying the equation. To avoid padding extra space to Weight and Threshold memories of a layer, P and S should be selected from the divisors of the number of rows and columns of measured total weight size of that layer, respectively. For a layer taking CC clock cycles to calculate all required activations, the FPS obtained will be

$$FPS = \frac{\text{Hardware Clock Rate}}{CC}. \tag{5}$$

Fig. 3 shows performance and memory utilisation (BRAMs and LUTs) of various FINN implementations on the ZC702 board when the network layers are balanced. Only the total PE count of all the layers (without SIMD count) is shown for each experiment for simplicity. The top plot illustrates the performance we expected from each experiment based on (3), (4), and (5) (or equivalently, using HLS Analysis Perspective report) and the one obtained in practice, and the lower plot shows the corresponding memory utilisation for each of them. The CIFAR-10 classification accuracy over the first 1000 test images using this network is 78.5%. As seen, for ZC702 board, the current BRAM utilisation for every possible configuration of this network is too high to allow any other design to be implemented alongside the classification design such as hardware that could extract regions of interest in a large HD frame and then scale to 32x32 sub-frames for use in CIFAR-10 network.

FINN places the network parameters (weight and threshold values) onto the OCM of FPGA for higher performance. Since every memory instance of over about 1 Kb is assigned to BRAMs by Vivado HLS tools (lower-capacity instances are assigned to LUTs and FFs), the FPGA device BRAMs are highly utilised by the network parameters. Furthermore, inter-layer stream buffers increase BRAM pressure too. Therefore the memory components available on board should be used efficiently, but as reported in [8], on average only $\sim 22\%$ of the storage space in the allocated BRAMs is actually used. One major contributor to this underutilization is how the compiler allocates BRAMs. For every memory allocation instance, BRAM utilisation is rounded to the next power of two for performance. Having multiple layers in a network, each with P instances of Weight memory and P instances of Threshold memory, a better allocation of BRAMs could reduce its under-utilisation by a large amount.

One possible method to reduce BRAM under-utilisation for each engine is to use HLS `array_partition` pragma to realise memory instances with desired capacities. Block-type array partitioning can be used in every layer of the network, for each of the P instances of Weight memories or P instances of Threshold memories, if the allocated BRAMs can be reduced by dividing that instance of memory into multiple smaller sized ones. This will prevent a large unused gap being appended to memory instances. Although this method could help with partitioning files taking up multiple BRAMs, the smaller files using only a fraction of one BRAM cannot be improved here. Applying efficient partitioning of memories to every configuration shown in Fig. 3, we achieve the results shown in Fig. 4 where BRAM utilisation drops 15%–18%. Block type array partitioning causes the configurations with higher PE counts to retain their original obtained performance but the ones with lower accelerations to slow down.

Since image classification designs are typically part of a bigger design in practice (e.g. used in live video streams), in this experiment we select the configuration with the lowest BRAM utilisation to release resources for other hardware blocks; the implementation reaching 430 images/second and utilising 65% of the ZC702 board BRAMs is used through the rest of this article for further development.

B. Decision-Making Unit (DMU)

The DMU is designed to estimate the success or failure of the single-bit network inference. The inputs to the DMU are the outputted class scores from the BNN, and the DMU output is a measure indicating whether the confidence value of the BNN classification is considered high or not. The DMU function is required to have these two characteristics: it should be trainable, and it should squash a vector of arbitrary values (the resulting BNN scores) to a real probability value in the range 0–1 (BNN success probability). A good DMU function candidate for the classification problem is Softmax. For this work, we executed the FINN classification on CIFAR-10 training dataset and created a new dataset composed of the FINN output scores and its identification result (1 indicating success and 0 failure). This dataset was used to train a Softmax layer with the 10 scores used as the input and the single identification result as the label. The resulting trained Softmax

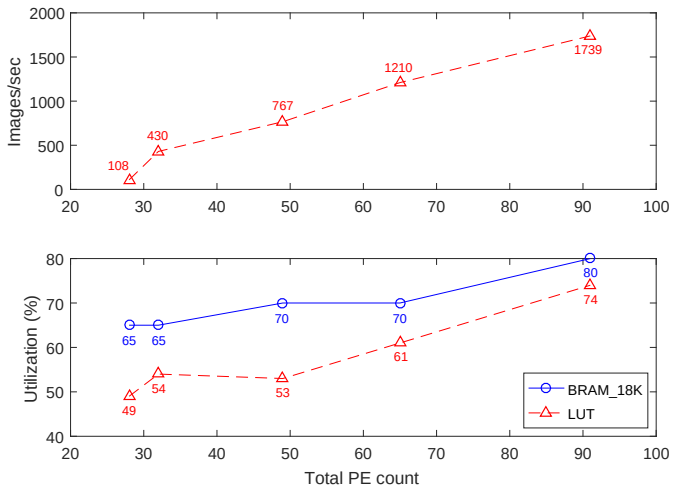

Fig. 4. Performance and area utilization for different CIFAR-10 implementations on ZC702 board each with a different degree of parallelism, when BRAM utilisation is made efficient by block type array partitioning.

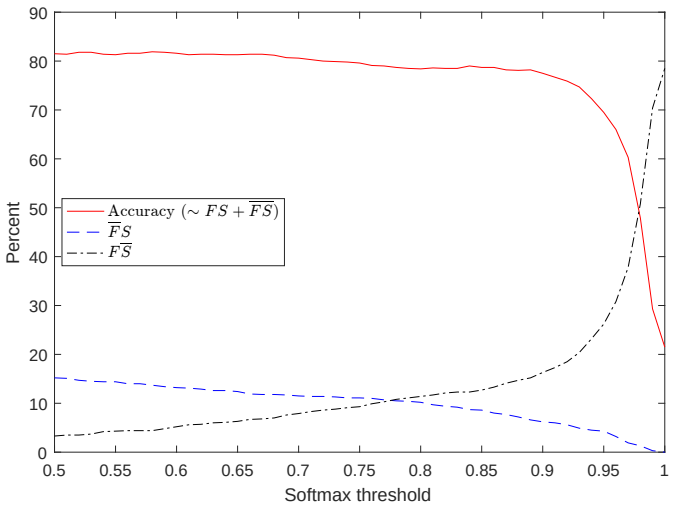

Fig. 5. Softmax layer accuracy and $\overline{F}S$ and $F\overline{S}$ values for different thresholds obtained when applied to CIFAR-10 training dataset.

layer can receive 10 FINN class scores of a test image and output one probability value estimating if it was a correct detection by FINN.

The images classified by FINN and then estimated by Softmax can fall in one of the following categories (represented as percent of total):

- FS: classified correctly by FINN and estimated as being correct by Softmax too;
- $\overline{FS}$: classified incorrectly by FINN and estimated as being incorrect by Softmax too;
- $\overline{F}S$: classified incorrectly by FINN but estimated as being correct by Softmax;
- $F\overline{S}$: classified correctly by FINN but estimated as being incorrect by Softmax.

The classification of the images in FS and $\overline{F}S$ will not be rerun in the host contrary to the ones in $F\overline{S}$ and $\overline{FS}$. The obtained Softmax accuracy is $FS + \overline{FS}$, with FS being the net contribution of FINN to the multi-precision system's classification accuracy. $\overline{F}S$ is of key importance here as this puts a cap on the achievable accuracy by the multi-precision system; as $\overline{F}S$ increases, the multi-precision system accuracy decreases. On the other hand, given a more accurate but slower classification model on the host system, as $F\overline{S}$ increases, the multi-precision system's classification speed decreases. To control $\overline{F}S$ and $F\overline{S}$, a threshold on Softmax estimation can be applied. To balance the multi-precision classification accuracy against the host classification speed, the threshold selection can be made considering the following relations

$$\overline{F}S \sim \frac{1}{\text{Host Classification Speed}}, \tag{6}$$

$$F\overline{S} \sim \text{Host Classification Speed}. \tag{7}$$

On CIFAR-10 training data, as shown in Fig. 5, in threshold values range of 0.5–1, $\overline{F}S$ decreases while $F\overline{S}$ increases. To strike a balance between the multi-precision classification accuracy and speed, a suitable threshold selection given a slow host classifier should be from the start of this range, and given a fast host classifier from the end of the range (up to 0.9). The threshold setting and the results obtained for this work is shown in Table II. With this setting, the maximum achievable multi-precision accuracy will be 91.3%.

TABLE II
SOFTMAX LAYER THRESHOLD SETTING AND THE OBTAINED VALUES FOR CIFAR-10 EXPERIMENT

Threshold	FS	$\overline{FS}$	$\overline{F}S$	$F\overline{S}$
0.84	66.2%	12.8%	8.7%	12.3%

In our work, every inference by the trained single-layer Softmax function consists of ten floating-point multiplications and their sum, a bias addition, and application of a Sigmoid positive transfer function. Softmax layer is directly added to the host C++ source code.

C. Floating-Point Network on ARM Host

The floating-point network in the CPU uses both cores but its classification rate is much slower than the FPGA BNN. The adjustable ratio of images to be re-processed by the high-accurate network in the host (25.1% in our experiment according to Table II) and the very fast but less accurate classification by BNN in FPGA are used to balance speed against accuracy.

For this work, the framework used for the higher accuracy network implementation in the host CPU is Caffe which is open source and written in C++. A strong point of Caffe for multi-core CNN implementations is its efficient and easy utilisation of OpenBLAS[2]. To achieve high performance for linear algebra operations on the ARM processor, we compiled and installed OpenBLAS with OPENMP on the ZC702 board. Although in our 32-bit ARMv7 processors OpenBLAS provides efficient CNN operations, it does not utilise the NEON architecture due to limited performance gains. OpenBLAS utilises the high-performance NEON (replaced by Advanced SIMD) in 64-bit ARMv8 processors, hence the performance

[2]http://www.openblas.net

TABLE III
THE THREE NETWORKS USED FOR CLASSIFICATION ON CIFAR-10 DATASET IN PROCESSING SYSTEM

Model A	**Model B**	**Model C**
input (32×32 RGB image)		
5×5-conv-32	5×5-conv-192 ReLU	dropout
pooling ReLU	1×1-conv-160 ReLU	3×3-conv-96 ReLU
LRN	1×1-conv-96 ReLU	3×3-conv-96 ReLU
5×5-conv-32 ReLU	pooling	3×3-conv-96 ReLU
pooling	dropout	dropout
LRN	5×5-conv-192 ReLU	3×3-conv-192 ReLU
5×5-conv-64 ReLU	1×1-conv-192 ReLU	3×3-conv-192 ReLU
pooling	1×1-conv-192 ReLU	3×3-conv-192 ReLU
FC-10	pooling	dropout
	dropout	3×3-conv-192 ReLU
	3×3-conv-192 ReLU	1×1-conv-192 ReLU
	1×1-conv-192 ReLU	1×1-conv-10 ReLU
	1×1-conv-10 ReLU	pooling
	pooling	

TABLE IV
PERFORMANCE OF NON-HETEROGENEOUS CIFAR-10 TESTING-DATA CLASSIFICATION, FOR MODELS A, B, AND C RUNNING IN ARM CPU OF ZC702 BOARD (FINN IS ALSO INCLUDED FOR COMPARISON)

	Model A	**Model B**	**Model C**	**FINN (FPGA)**
Accuracy	81.4%	89.3%	90.7%	78.5%
Images/sec	29.68	3.63	3.09	430.15

of experiments in this article can be further improved in these high-end processors.

We test three Caffe models for CIFAR-10 classification in this work: Model A which is built based on Alex Krizhevsky's cuda-convnet[3], Model B which is the "Network in Network" model described in [9], and Model C which is an "ALL Convolutional Net" model described in [10]. The networks of these models are shown in Table III. Model A is a fast classifier and better suited for slow processors such as our dual core ARM Cortex-A9, compared to Models B and C which are deeper. Table IV shows the testing accuracy and classification rates of these models executed in host when FPGA is left idle (FPGA-based FINN classification results are shown for comparison). These results verify that while the single-bit precision FINN provides much faster inference, its accuracy falls short of even a simple floating-point network such as Model A.

D. Multi-Precision Results

The accuracy/speed balanced results obtained by parallel classification of CIFAR-10 test dataset on FPGA and host (for different host Model selections) are listed in Table V. Of the three implementations, the combination of Model A and FINN increases FINN accuracy by 4% (and even increases the accuracy of the floating-point network of host by 1.1%) and the resulting 90.82 classification rate is well above 60 fps required in most real-time video streaming applications. The deeper Models B and C improve the multi-precision accuracy by a larger amount but would require faster host CPU for real-time applications. In these experiments, the host accuracies for Models A, B, and C on the subset of images selected by the Softmax function are 65%, 79%, and 83% respectively; compared to the non-heterogeneous accuracy results in Table IV, this shows that the re-inference process in host is applied to hard-to-classify subset of images.

[3]https://code.google.com/archive/p/cuda-convnet/

TABLE V
PERFORMANCE OF HETEROGENEOUS MULTI-PRECISION CIFAR-10 TEST DATASET CLASSIFICATION, FOR MODELS A, B, AND C RUNNING IN ARM CPU IN PARALLEL WITH FINN IN FPGA OF ZC702 BOARD

	Model A & FINN	**Model B & FINN**	**Model C & FINN**
Accuracy	82.5%	86%	87%
Images/sec	90.82	14.00	11.98

IV. CONCLUSION

In this article, we proposed a multi-precision convolutional neural network inference approach in which the full dataset and a subset of it are respectively run in a reduced and full precision implementations in parallel. A DMU function is used to set a trade-off between accuracy and speed by deciding on the degree of subset recalculation. The results show that the multi-precision concept can provide overall balanced results improving BNN accuracy and floating-point network throughput. The results in the tested configuration are limited by the overall low throughput achieved in the weak Cortex A9 processors. Our future work aims at using the multi-precision concept on higher-end heterogeneous devices that incorporate ARMv8 processors with active NEON engines and also considering use of mixed precision on the FPGA hardware as well.

ACKNOWLEDGMENT

The authors would like to thank Xilinx Corporation for providing FINN framework and helping with its implementation.

REFERENCES

[1] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[2] M. Courbariaux and Y. Bengio, "BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016.

[3] Y. Umuroglu *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: ACM, 2017, pp. 65–74.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding," *CoRR*, vol. abs/1510.00149, 2015.

[6] F. N. Iandola *et al.*, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *CoRR*, vol. abs/1602.07360, 2016.

[7] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, G. Lorette, Ed., Université de Rennes 1. La Baule (France): Suvisoft, Oct. 2006.

[8] N. J. Fraser *et al.*, "Scaling binarized neural networks on reconfigurable logic," *CoRR*, vol. abs/1701.03400, 2017.

[9] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[10] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014.